Remarks

Claims 1-48 are pending in this application. New claims 23-48 have been added. The new and amended claims are fully supported by the specification. No new matter has been added.

Claim 22 was amended so this claim should no longer in a form which is subject to examiner's section 101 rejection.

Claim Rejections Under 35 U.S.C. § 102 and § 103

Claims 1-5, 7-8, 11-18 and 21-22 are rejected under section 102(b) as being anticipated by Linnermark (US Patent 5,594,904). Claims 6, 9, 10, 19, and 20 were rejected under section 103. Reconsideration of the rejections and allowance of the claims are respectfully requested in view of the above amendments and comments below.

There are numerous fundamental differences between the present invention and the invention described in Linnermark (US Patent 5,594,904). The present invention describes an effective means to trace a path of code execution in servicing a request within each tier and across different tiers of a multi-tier heterogeneous distributed data processing system. In contrast, the invention described in Linnermark was designed to trace a path traversed by a message (or a signal) within a message passing system, which is something very different from the invention.

The following are quotes from the Linnermark specification highlighting the invention's limitation to message tracing.

> "The processor's 34 software performs many tasks, including those identified above, which are initiated by a software signal or software message, i.e., a software instruction including relevant data unique to the task. Software signals or

software messages can be traced as part of the diagnostics being performed by the SPC exchange 30." (Column 2, line numbers 39-44)

"When thread-tracing commences, a key 49 is attached to all software signals or messages sent during execution." (Column 6, line numbers 33-34)

"But if in addition to process tracing the key structure is assigned to all software signals or messages sent from a process, this is thread tracing. When another process receives such a message, the key structure is assigned to the receiving structure and activated when the receiving process is activated." (Column 7, line numbers 2-7).

"The trace-thread 50 propagates between the processes 51-55 by means of software signals or messages 56-59." (Column 7, line numbers 30-32)

"The processes 51-55 which form the trace-thread 50 are linked by the messages 56-59." (Column 7, line numbers 44-45)

"However, the first and second daemon in the half call process 54 on the terminating side B have been activated, as indicated by the solid circles, by a key carried on the message sent to the traffic control process 54." (Column 7, line numbers 60-64)

"If the key value contained in the message 115A fits the lock stored in the daemon D5, thread tracing will occur as represented by the solid arrows for the trace thread 115A/116A/117A/118A and relevant data will be stored. If, however, the key value obtained in the message for the second call B does not fit the lock of the daemon D5, there will be no thread tracing activity as represented by the dashed arrows for the trace thread 115B/116B/117B/118B." (Column 10, line numbers 54-62)

Application Number 10/657,770

Page 14/19

"Thus, a single key connected to a message will open the lock for all of the daemons in that group." (Column 12, line numbers 7-9)

Linnermark specification assumes an application environment being a messaging system (e.g., a phone system). Quite the contrary, the intended application environment of the present invention is a general purpose data processing system.

In additional to the difference between application environments, further distinctions will become obvious upon a detailed comparison of the two inventions.

First, there are two implicit assumptions in the Linnermark specification. The first assumption is that the invention relies on a system it traces to provide a carrier (e.g., a message) for the key it used in thread tracing (as discussed above). In addition, the scope of a trace is limited to the path traversed by the carrier. The second assumption is a message that a user wants to trace can be modified to hold a key, and the inclusion of the key will not break normal operation of the system (or else the system being traced has to be modified to handle a key). The Linnermark specification does not suggest any non-intrusive means to insert a key into a message on an existing messaging system that does not affect normal operation of the system.

On the other hand, as recited in for example, claims 1, 16, 22, 36, 43, and 46, the present invention is designed to trace a path of code execution that services a request as the request enters a data processing system (e.g., a HTTP request). Servicing such request may involve one or more application programs running on one or more computers (e.g., a web server, an application server and a database server) where a uniform message passing system does not exist. Tracing a path of code execution in such environment may involve interception at multiple points within a process on a computer. Interception may also occur on many computers that participate in servicing the request.

In order to trace a path of code execution in such complex environment, a number of challenges must be overcome. The challenges that are addressed by the present invention

Application Number 10/657,770   Page 15/19

include: 1) ability to non-intrusively install interceptors into an application program and other layers of an application stack to serve the tracing needs; 2) ability to accurately correlate data collected across different tiers of a heterogeneous computing environment traversed by an execution path; 3) ability to reconstruct an execution path based on collected data; 4) ability to handle loops and recursion on an execution path; and 5) ability to accurately identify parallel code execution paths on an execution path. The present invention provides a solution based on a tracing token. A tracing token serves both as an activator during tracing phase and a correlation identifier in post tracing phase. More importantly, the present invention describes effective techniques of relaying a tracing token both in-process and out-of-process to enable tracing a path of code execution.

The prior art does not show or suggest the features recited in claims 1, 16, 22, 36, 43, and 46. Therefore, these claims should be allowable for at least these reasons. Claims 2-15, 17-21, 23-35, 37-42, 44-45, and 47-48 are dependent on one of the independent claims and should be allowable for at least similar reasons.

Second, Linnermark describes a method of using a key to activate message tracing at a daemon wherein the key is inserted into a message as directed by a user. The assumption is that as long as a system that handles routing and processing of the message supports having a key in the message (i.e., adding the key to the message does not affect normal operation of the system), the task of relaying a key from one daemon to another daemon is automatic. According to Linnermark, thread tracing relies on a key presents in a message matching a lock presents on a daemon. This means tracing code execution in a code module that has no access to the key is not possible. For example, if a library function that validates a field in a message which has access only to the field it validates, the function will not be traced until the library is modified to make access to the key available to the function. This is because a daemon install in the library function is not given a key to open a lock.

Application Number 10/657,770   Page 16/19

In the contrary, the present invention describes provisions on relaying a tracing token within a process non-intrusively using shared memory; and relaying a tracing token between two processes by attaching a tracking token to a message non-intrusively or inserting a tracing token to a stream. Once a tracing token is placed in a shared memory location, any code fragment (or function) that has access to the share memory location can be instrumented with an interceptor and tracing will automatically extend to such code fragment. This means practically all functions within an application program, libraries, and operating system can be traced. The message mentioned here and in the specification of the present invention is an inter-process communication message which is typically point-to-point and is not routed. Whereas messages described in Linnermark are routed, and behave like network packets or e-mails.

The prior art does not show or suggest these additional features recited in claims 1, 16, 22, 36, 43, and 46. Therefore, these claims should be additionally allowable for at least these reasons. Claims 2-15, 17-21, 23-35, 37-42, 44-45, and 47-48 are dependent on one of the independent claims and should be allowable for at least similar reasons.

Third, Linnermark assigns a unique identifier to each daemon and describes methods to resolve identifier conflict in a large scale implementation. By making each daemon uniquely identifiable, a management application can communicate with a particular daemon to update locks on the particular daemon and activate the particular daemon.

The present invention does not assign a unique identifier to each interceptor. On the other hand, a unique identifier is assigned to each request being traced. By assigning a unique identifier to each request being traced and collecting the unique identifier along with other data at each interceptor the request has traversed, the data collected for the request can be uniquely identified. For example, the data collected may come from two application programs on two computers that the request has visited, two simultaneously running threads on the same computer serving the request, or a long running request that spans days instead of seconds.

Application Number 10/657,770   Page 17/19

The prior art does not show or suggest these additional features recited in claims 1, 16, 22, 36, 43, and 46. Therefore, these claims should be additionally allowable for at least these reasons. Claims 2-15, 17-21, 23-35, 37-42, 44-45, and 47-48 are dependent on one of the independent claims and should be allowable for at least similar reasons.

Fourth, Linnermark describe a complex key and lock system to support thread tracing. The system requires a user to configure locks on each daemon (or each group of daemon) that the user wants to trace using a trace tool and issue commands to cause a key (or a key structure) to be inserted into a message whereby activating thread tracing.

The present invention is based on a tracing token that gets relayed within a process and across processes and activating data collection at interceptors along a code execution path with no configuration required. The tracing token also serves as a correlation identifier allowing data collected along an execution path serving a request to be identified.

The prior art does not show or suggest these additional features recited in claims 1, 16, 22, 36, 43, and 46. Therefore, these claims should be additionally allowable for at least these reasons. Claims 2-15, 17-21, 23-35, 37-42, 44-45, and 47-48 are dependent on one of the independent claims and should be allowable for at least similar reasons.

Fifth, Linnermark describes a daemon group which is a collection of daemons where each daemon in the group is assigned a common lock. Since daemons in a daemon group have the same common lock, a message containing a key that matches such common lock activates any daemon in the daemon group. The key used to activate data collection at a daemon is unique within the system but not unique across difference traces. Therefore the key cannot be used to correlate data of an individual trace.

As described in the above, the present invention assigns a unique identifier to each request being traced. When data is collected at an interceptor, the unique identifier is recorded along with the data collected. The unique identifier serves as a correlation identifier (or key) during reconstruction of an execution path.

Application Number 10/657,770   Page 18/19

The prior art does not show or suggest these additional features recited in claims 1, 16, 22, 36, 43, and 46. Therefore, these claims should be additionally allowable for at least these reasons. Claims 2-15, 17-21, 23-35, 37-42, 44-45, and 47-48 are dependent on one of the independent claims and should be allowable for at least similar reasons.

Sixth, Linnermark describes a system that has one type of daemon that can be programmed to perform different tasks related to tracing a message path.
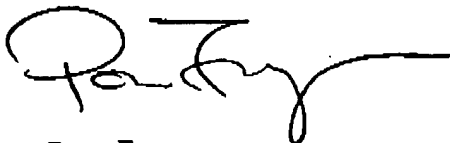
Whereas, the present invention describes a system with two distinct types of interceptors each providing a separation function, and cooperatively tracing a code execution path in an application program or a complex environment comprises dissimilar application programs.

The prior art does not show or suggest these additional features recited in claims 1, 16, 22, 36, 43, and 46. Therefore, these claims should be additionally allowable for at least these reasons. Claims 2-15, 17-21, 23-35, 37-42, 44-45, and 47-48 are dependent on one of the independent claims and should be allowable for at least similar reasons.

Conclusion

For at least the above reasons, applicant believes all claims are allowable.

Respectfully submitted,

Poon Fung
20375 Via Volante
Cupertino, CA 95014
(408) 777-8816

Application Number 10/657,770   Page 19/19